A Voucher-Based Security Middleware for Secure Business Process Outsourcing

Emad Heydari Beni[1(✉)], Bert Lagaisse[1], Ren Zhang[2], Danny De Cock[2], Filipe Beato[2], and Wouter Joosen[1]

[1] imec-Distrinet, KU Leuven, Leuven, Belgium
{emad.heydaribeni,bert.lagaisse,wouter.joosen}@cs.kuleuven.be
[2] imec-COSIC, KU Leuven, Leuven, Belgium
{ren.zhang,danny.decock,filipe.beato}@esat.kuleuven.be

Abstract. Business Process Outsourcing (BPO) enables the delegation of entire business processes to third party providers. Such scenarios involve communication between federated and heterogeneous workflow engines. However, state-of-the-art workflow engines fall short of a distributed authorisation mechanism for this heterogeneous, federated BPO setting.

In a cross-organisational context, the security requirements involve (i) delegation and verification of privileges in a confidential manner, (ii) secure asynchronous operations during the long-term workflows even when the users are logged-off, and (iii) controlling access to interfaces of the different workflow engines involved.

To address these challenges, we present a voucher-based authorisation architecture and middleware. We extended the WF-Interop [2] middleware with a security module to support this authorisation architecture. We further validated our contributions by prototyping a billing workflow case study on top of the extended WF-Interop middleware and evaluated the performance overhead of the security extensions to the middleware.

Keywords: Security middleware · Authorization · Business process

1 Introduction

Software service providers are evolving towards a Business Process Outsourcing (BPO) model. BPO refers to delegation of entire business processes to third party providers. Providers take over the complete business functions and are free to choose the implementations; consumers only receive the results of processes [2, 6, 11, 25].

For example, accounting departments of many companies outsource their billing processes to external service providers. These processes comprise activities such as documenting, shipment and payment. For instance, (1) an accounting manager submits an order to send some bills by the end of the month. (2) The billing process gets started at the provider's side by the accounting department. The service provider takes over the entire process. (3) They print and package

E. Bodden et al. (Eds.): ESSoS 2017, LNCS 10379, pp. 19–35, 2017.
DOI: 10.1007/978-3-319-62105-0_2

the bills. Afterwards, (4) they start a delivery process at a shipping company. The accounting department periodically inspects the running processes at the provider for progress updates to know the current status of the bills (e.g. sent, resent, paid, etc.).

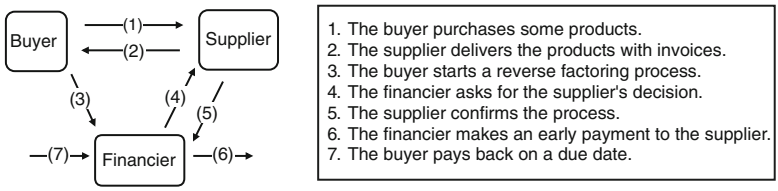

Fig. 1. Reverse Factoring (RF)

Reverse Factoring (RF) in the FinTech sector is another example of such a process. Reverse Factoring enables companies to pay their bills on time with assistance of financiers (also called brokers), see Fig. 1. In brief, (1) a buyer purchases some products from a supplier; (2) the supplier sends a bill to the buyer with a due date; the buyer wants to pay the bill on time to ensure their business continuity; (3) they therefore request a financier to get financial assistance; (4, 5) the financier evaluates the request in first place; then they negotiate with the supplier for their decision. If all parties reach an agreement on this process, (6) the financier pays the buyer's bill before its due date; and instead, (7) the buyer will pay the bill to the financier with an extended due date, perhaps with interest. Buyer companies tend to employ BPO in order to outsource the entire process to a provider (a broker/financier) to be able to concentrate on their core business. The Reverse Factoring scenario is the running example in this section.

Each of the business processes gets executed over different workflow engines located in different companies. These heterogeneous workflow engines have their own business-specific security mechanisms to protect their sensitive data and control the access rights. The diversity of technologies used in workflow management systems across the BPO parties introduces interoperability issues with respect to service computing in general, and security in particular. Heydari et al. [2] outlines the common patterns among such BPO scenarios as follows.

- Multiple parties are involved in BPO model, resulting in federated, heterogeneous workflow engines.
- Outsourced processes are long running workflows, e.g. taking days or weeks to be completed.
- Occasionally, the client parties inspect the progress of outsourced processes.

On the one hand, most of the security mechanisms in workflow engines are used to meet the intra-organisational requirements, e.g. Role Based Access Control (RBAC) [9]. On the other hand, the federated, cross-organisational approaches in more general context (e.g. WS-Trust [19]) do not particularly take

all of the process outsourcing characteristics into account. Such characteristics include hierarchical and iterative delegation of privileges, human-involvement, organisational confidentiality, long-running workflows or asynchronous operations without active sessions (see Fig. 2).

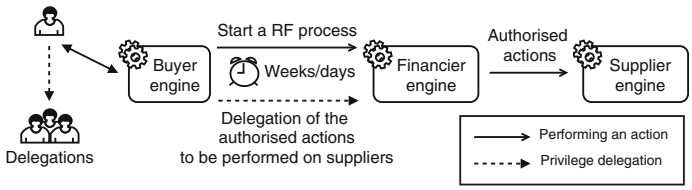

Fig. 2. Reverse Factoring (RF)

The scope and contribution of this paper is a security architecture and middleware that support following authorisation requirements and features in a BPO setting with federated, heterogeneous workflow engines.

1. *Delegation and validation of all or subsets of authorised privileges (access rights) to other involved people or executing, federated workflow engines.* For example, an accounting manager (buyer company) delegates the progress inspection privilege to an accountant; or an accounting workflow engine delegates a subset of its BPO functional rights (e.g. starting a process at a seller) to a financier engine.
2. *Secure asynchronous operations during long-term workflows even when the users are logged off.* For example, an accounting engine in a buyer company may require to perform an action (e.g. start/inspect a process) when the accountant is not logged in and there is no actual session available in the execution context of the engine. Therefore, there should be a way for the buyer's engine to authenticate against the financier's engine and perform the authorised actions.
3. *Controlling access to interfaces of workflow engines in the context of BPO.* For example, when an accounting manager `starts` a Reverse Factoring process, an accountant should not be able to `cancel` the process, but only inspect it. That constraint should be reflected to the accounting workflow engine by the application interface (API) of the financier's workflow engine.

To address these requirements, we present an access and enforcement mechanism as an integral part of the BPO model and middleware. This system works with *vouchers* (also called *tokens* or *assertions*), i.e. a digital representation of a claim or set of claims which has been certified by a particular entity [22]. Vouchers establish a decentralised authorisation management that aims to provide trust and security assurance to the involved parties in BPO scenarios.

To validate the voucher system, we implemented a security module for the *WF-Interop* middleware [2], i.e. a middleware interfacing heterogeneous and federated workflow engines in a unified RESTful architecture to support the BPO

use cases. The security module enables BPO consumers and providers (1) to manage (i.e. produce or delegate) security vouchers and (2) to verify the privileges and integrity of vouchers for each service call.

Furthermore, WF-Interop has a *hypermedia-driven* application interface, meaning that it enables the service consumers to discover the capabilities of underlying workflow engines by offering pointers to the next possible actions upon each service invocation. For example, if a consumer, via WF-Interop, starts a workflow instance in a Ruote [17] engine, WF-Interop embeds hyperlinks of other related actions such as `pause` and `abort` in the response. The capability propositions are based on the underlying engine, the type, the state of the process, and most importantly, the access right of the entity identity. **Hence the security module securely controls the consumer-specific action propositions based on the voucher content.**

The rest of this paper is structured as follows. Section 2 describes the necessary background about business processes and BPO, as well as an overview of related work in the authorisation domain. Section 3 introduces a voucher-based architecture to achieve secure BPO. Section 4 validates the aforementioned concepts and the security extensions to the WF-Interop middleware by a billing workflow case study. Section 5 evaluates the WF-Interop extensions in terms of performance. Section 6 concludes this paper.

2 Background and Related Work

In this section, we present a brief overview of workflows[1] and Business Process Outsourcing (BPO). Afterwards, the WF-Interop [2] middleware is described, including the key interfaces and hypermedia-driven architecture. In addition, the related work in the domain of security protocols and frameworks for authentication and authorisation are presented.

2.1 Background on BPO and WF-Interop

A business process is a group of activities that, once completed, will accomplish an organisational goal. For example, when you purchase a product online, you start a business process of purchase. This business process contains activities such as order placement, bank transfer, inventory checks and shipment. Once all are completed, you receive the product (the main goal).

A *process definition* is a representation of what is intended to happen [5], described by a business process modelling language such as BPMN. It contains a sequence of activities showing the order, relationships and semantics of the business process. Workflow engines execute the activities of process definitions. For each round of execution, an instance of a definition is created, holding a set of context-specific variables. *Business Process Outsourcing (BPO)* refers to delegation of entire business process to third party providers. Process definitions get deployed, instantiated and then executed entirely by different workflow engines.

[1] In this paper, we use the terms *business process* and *workflow* interchangeably.

WF-Interop is a middleware interfacing heterogeneous and federated workflow engines in a unified RESTful architecture aiming at facilitating BPO. Heydari et al. [2] describes WF-Interop, which has three interfaces: (i) *deployment*, i.e. enabling consumers to manage process definitions; (ii) *activation*, i.e. enabling consumers to activate process instances; (iii) *progress monitoring*, i.e., enabling them to monitor the progress of running instances. Accordingly, all workflow activities are delegated to the third parties and the level of communication of BPO clients is limited to coarse-grained interactions provided by the interfaces.

Hypermedia-driven interfaces in BPO. WF-Interop interfaces leverage well-known principles such as Hypermedia as the Engine of Application State (HATEOAS). In brief, when a BPO consumer calls a service function from one of the WF-Interop interfaces, WF-Interop embeds some navigational information (a set of links) in the response. For example, the Ruote workflow engine supports a set of functionalities on process instances such as `start`, `get`, `pause`, `resume` and `abort`. A BPO consumer calls the `start` functionality (from the `activation` interface) of WF-Interop. The WF-Interop middleware acts as a facade to several workflow engines and provides a uniform, coherent abstraction for consumers. Therefore, for this request it uses the built-in adapter for Ruote and starts a process. The process instantiation is an asynchronous service call, meaning that WF-Interop responds to the request initiator earlier than the complete process instantiation. In the body of the response, it embeds {`get`, `abort`} as the relevant navigational links. If the consumer calls the `get` function for that process instance after a while (when the process is instantiated), WF-Interop proposes {`get`, `abort`, `pause`} in the response. The capability propositions of WF-Interop are based on the underlying engine, the type, and the state of the process.

2.2 Related Work

Workflow engines support different access control models. Role-Based Access Control (RBAC) [9] is an access control mechanism that comprises users' roles and privileges. Attribute-Based Access Control (ABAC) [13] employs a more flexible paradigm by use of policies combining attributes and producing boolean logic outcomes. Business processes benefit from these mechanisms for authorisation of users' actions and restricting the access to resources in an intra-organisational context [3,23]. WS-HumanTask [10] has an emphasis on the human-involvement in business processes by providing roles illustrating actions that users can perform on tasks. Other approaches to access control for resources in a workflow context are studied in [18]. Considering the delegation of roles and access rights, some studies presented delegation models and frameworks for RBAC [24,26]; moreover, an extensive comparison of delegation models in a business context is provided in this review [21]. Most of these works did not take the cross-organisational characteristics of business process outsourcing into account.

WS-Trust [19] introduces a Security Token Service responsible for issuing tokens. It establishes a broker trust relationship among participants involved in

distributed systems. Recently the state of practice has moved towards the OAuth 2.0 authorisation framework [12]. Through different flows, OAuth2 enables third party applications to have limited authorised access to services on behalf of the resource owners by providing access tokens. To harden the authorisation mechanism presented in OAuth2, OpenID Connect [20] adds an identity layer on top of the OAuth2 protocol in order to provide authentication.

Another improvement is to employ Macaroons [4] to structure each OAuth2 token. Macaroons embed caveats (i.e. that defines specific authorisation requirements), as well as attenuation and contextual confinement of authorisation requests. The proof-carrying characteristic of Macaroons is based on an HMAC-based construction inspired by Merkle-Damgård hash function.

In the public-key-based area, SPKI/SDSI [8] employs name and namespace certificates to define identities, and authorisation certificates (delegatable by subjects) to define what each principal is allowed to do. To perform an action on a secure api, a certificate chain needs to be provided by the subjects.

All of the given mechanisms focus on authorisation and authentication of parties in a generic context which can be applied to business processes in general and BPO in particular. But most of the mechanisms support some or none of the cross-organisational characteristics of a BPO context with federated workflow engines. For example, the lacking characteristics include hierarchical structure of authorisation, confidential assertions, long-running workflows with asynchronous operations, and secure proposition filtering of HATEOAS which is used in the BPO middleware.

3 Secure BPO

In this section, we describe a security architecture and middleware for the three requirements defined in Sect. 1:

1. Delegation and validation of access rights using a voucher-based approach.
2. Secure asynchronous operations by workflow engines when the executing user is logged-off.
3. Secure filtering of HATEOAS propositions in BPO APIs.

The features of this security architecture are implemented as extension to the WF-Interop middleware and are applied to the communication between heterogeneous and federated workflow engines. In Sect. 3.1, we present the voucher structure, as well as how it facilitates the delegation and verification of privileges. In Sect. 3.2, the secure asynchronous operations are described. The secure filtering of HATEOAS propositions is explained in Sect. 3.3.

3.1 Delegation and Validation of Access Rights

We present a voucher-based authentication and authorisation protocol that establishes an architecture aiming to provide delegation and validation of access

rights in BPO scenarios. In this subsection, we describe (i) the structure of a voucher; (ii) the procedure of voucher delegations; (iii) the voucher verification steps; and lastly (iv) the renewal procedures.

Voucher Structure. *"A claim is a statement that something is the case, without being able to give proof"* [1]. A voucher (also called *token* or *assertion*) is a digital representation of a claim or set of claims which has been certified by a particular entity[2] [22]. In addition to the *non-repudiation*[3] characteristic common among similar works [15], our vouchers contribute a set of features:

- *Hierarchical structure.* A voucher owner can delegate a subset of his claims to a new subject by creating a new voucher with less or equal validity period. The parent vouchers are embedded within a child voucher. Therefore, the verifiable iteration of parents provides a chain of trust.
- *Confidentiality.* The subject cannot learn anything from the parent vouchers because all of the parent vouchers are protected by hybrid encryption[4].
- *Stateless.* The identity provider and servers store no information about sessions and delegations (parent vouchers).

According to the JSON web tokens (JWT) [15] representation, there are three types of claims: registered, public and private.

Listing 1.1. Voucher payload

```
{
    parent: Enc(parent vouchers),
    jti: a voucher unique identifier,
    iss: the issuer identity,
    sub: the subject identity,
    iat: the issuing time,
    exp: the expiration time
    wfi: the workflow identifier,
    wei: the engine identifier,
    actions: the list of allowed actions
}
```

The registered and public claims are predefined in the standard [15], e.g. `jti`, `iss`, `sub`, `iat` and `exp` (see Listing 1.1). We extended the representation by adding extra fields such as `parent`, `wfi`, `wei` and `actions` in the form of private claims. The `parent` field is the issuer's voucher encrypted using a hybrid encryption scheme with the public key of an identity provider (WF-Interop). The `wfi` and `wei` are the unique identifiers of the workflow and the responsible

[2] In BPO, entity can be a person, a group, a department or a workflow engine. Attributes of an entity can be an email address, a public key or a randomly generated value.

[3] Using cryptographic signatures the integrity of the voucher and the authenticity of the issuer are guaranteed.

[4] This scheme is a mix of a public-key cryptosystem with a symmetric-key crypto system, e.g. OpenPGP.

workflow engine. Lastly, the `actions` field is a list of permitted actions that the subject is able to execute.

The JWT representation [15] has three segments: a header, a payload and a signature. Listing 1.1 represents the payload of a voucher. The header describes the cryptographic operations applied to the JWT [14] token, e.g. the scheme used in signatures. The third segment is the cryptographic signature as the vouchers are secured.

In summary, we employed the JWT standard representation with extension of embedded parent vouchers and BPO related fields as private claims.

Voucher Delegations. In this subsection, we describe an approach for delegation of vouchers. The key differentiators are decentralised voucher generation, iterative delegations and user-owned cryptographic keys.

In our running example, we use fictional characters: Alice, Bob and Carol. Each of the actors owns a pair of public-key cryptographic keys[5] (pk, sk) for voucher-based functionalities and a secret for authentication against WF-Interop. WF-Interop is aware of the public keys (Pk's), the secrets, the workflow engines, and the workflows with all set of possible actions. The public keys of the users are known to WF-Interop, however the private keys are only known to the users. In addition, WF-Interop has its own pair of public and private keys. The public key is broadcasted to all actors. Figure 3 illustrates voucher creation, voucher usage and voucher delegation:

1. *Voucher creation.* Alice wants to create a voucher for Bob. (i) She creates a voucher V_{ab} by adding some claims including permitted actions (e.g. starting a payment process), and (ii) signs it using her Sk. (iii) Afterwards, she sends the voucher to Bob via WF-Interop over a secure channel. WF-Interop checks the identifier of the voucher because the voucher might have been revoked by the issuer. In other words, WF-Interop only stores the identifiers of revoked vouchers.
2. *Voucher usage.* Bob wants to perform an action on a secure workflow engine which is authorised by Alice using the voucher V_{ab}. (i) He sends the execution request to the engine via WF-Interop along with his secret and the voucher. Before execution, (ii) WF-Interop verifies the authenticity of V_{ab} and validates that whether Bob's claim to perform the action is available in the voucher (refer to Sect. 3.1). (iii) Then it executes the actual request.
3. *Voucher Delegation.* Bob wants to delegate a subset of the permitted actions from his voucher V_{ab} to Carol. (i) He creates a voucher V_{bc} and adds some of his claims to it. (ii) He encrypts the V_{ab} with the public key of WF-Interop using hybrid encryption and embeds it in the V_{bc} as the parent voucher. His *confidentiality* is protected as she is not able to see the complete set of his claims within V_{ab}. (iii) Afterwards, he sends V_{bc} to Carol via WF-Interop over a secure channel. The validity period of the vouchers are less than (or equal to) the validity of the parent vouchers.

[5] Using a public key (pk), one can either encrypt a message or verify a signature and with a private key (sk) one can either decrypt or sign a message.

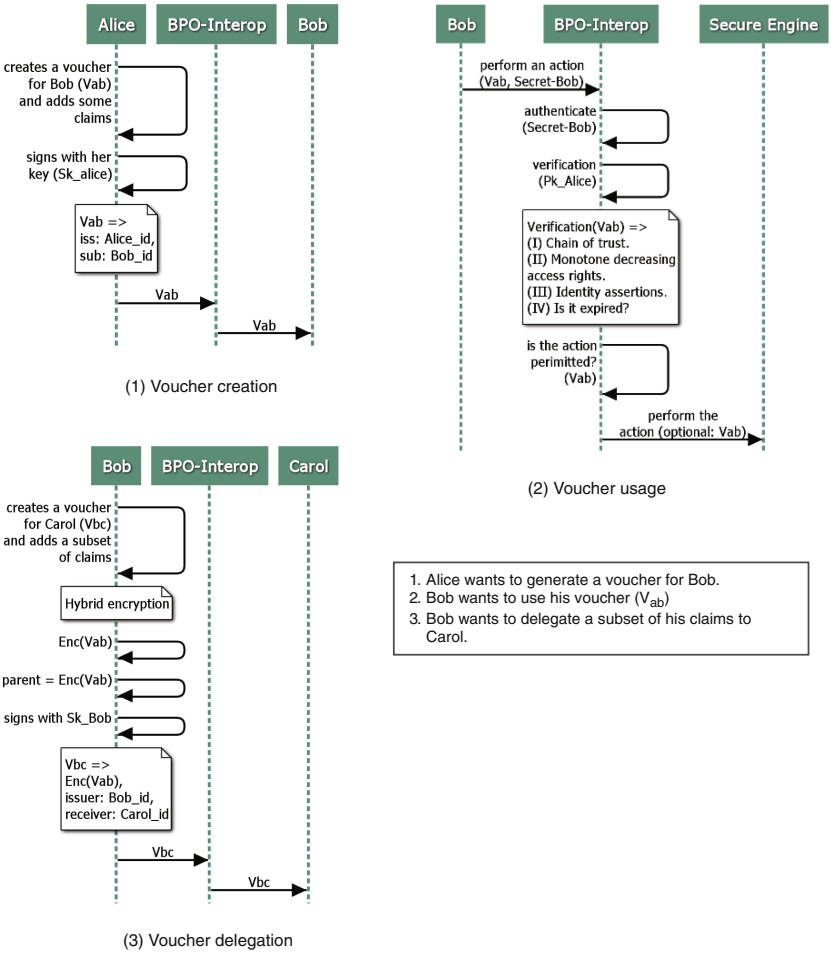

Fig. 3. Voucher creation, usage, and delegation

In a BPO context, iterative and hierarchical delegation of vouchers is unavoidable. For example, an accounting director may delegate her executive tasks to the other employees; or a BPO provider engine may need to again outsource some part of the process to another provider's engine for special services.

Voucher Verifications. The WF-Interop security component verifies the validity of the vouchers upon usage (e.g. see Fig. 3 when Bob wants to use his voucher). Obviously, a voucher must contain the claim to access a resource that the request initiator wants to have access to. Besides, the verification procedure encompasses three more criteria:

- *Chain of trust.* WF-Interop is able to decrypt the embedded parent vouchers and verify the signatures. The issuer of the child voucher must be the subject of the parent voucher.
- *Monotone decreasing access rights.* Every child voucher must contain less/equal claims than/to the chain of parents, meaning that there must be no unknown claim in the set of claims.
- *Identity assertions.* The identity of the issuer must be the subject of the parent voucher. Moreover, the provided secret adds a layer of authentication to the verification of the subject.

Ultimately if the verification process succeeds, the delegatee gains access to the requested resource on behalf of the delegator, e.g. starting an invoice delivery process by Carol.

Voucher Renewal. Vouchers have a validity period which is set by the issuer. A BPO provider's voucher may expire before completing the business process, e.g. a problem in the delivery of a bill may slow down a billing process. In such a case, the subject requests the issuer(s) to renew the voucher.

Assume that Carol wants her voucher (V_{bc}) to be renewed. V_{bc} is issued by Bob; and V_{ab}, as the parent of V_{bc}, is issued by Alice iteratively. (1) Carol sends a renewal request to WF-Interop along with V_{bc} and her secret; (2) WF-Interop authenticates Alice using the provided secret; (3) then it recursively checks the expiration time of the parents' vouchers (V_{ab}). If Bob's V_{ab}, as the first parent voucher, is still valid, it sends a renewal request to him. Otherwise, the request will be sent to Alice. (4) The responsible issuer creates a new voucher and sends it back to Carol via WF-Interop.

3.2 Secure Asynchronous Operations

Sometimes workflow engines perform BPO actions on external workflow engines when the responsible user is not logged in. For example, there might be a timer in the business process to start an instance at the service provider in the future; or the client process inspects the progress of the outsourced workflow periodically every 2 h; or the logged-in person is not the person with the right authority to continue the execution. That means that the subject has no authenticated session in the workflow engine at the time of a service invocation because the outsourced processes are typically long running; in other words, the client workflow engine cannot authenticate against the BPO provider engine to perform the authorised actions.

There are two options to address this issue using vouchers. (i) The workflow engine keeps the vouchers in the running process instance and impersonates the subject at the execution time. In a more restricted case, (ii) the responsible person may delegate a set of his access rights by creating a new child voucher to the executing workflow engine or a person in charge for a specific amount of time as described in Sect. 3.1.

3.3 Secure HATEOAS Filtering

As described in Sect. 2.1, WF-Interop comprises a set of hypermedia-driven interfaces which are based on a principle called *Hypermedia as the Engine of Application State (HATEOAS)* [2]. In brief, when a client accesses a resource at the provider's workflow engine, WF-Interop proposes a set of related resources to the request initiator based on the (1) underlying workflow engine, (2) the type and (3) the state of the running process instance. For example, when an accounting manager `starts` a Reverse Factoring process, the accounting workflow engine receives navigational information of the related resources such as `pause`, `inspect` and `cancel`. In this case, an accountant should only be able to `inspect` the progress without being able to `cancel` the process.

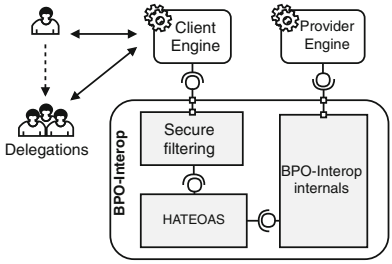

Fig. 4. Secure HATEOAS filtering

As illustrated in Fig. 4, the HATEOAS component retrieves the related resources from the storage mechanism of WF-Interop upon each action execution on the provider's workflow engine. Clients enable the secure filtering component to filter out the unauthorised resources by passing along their vouchers to the BPO interfaces of WF-Interop. In other words, a client only receives a set of permitted resources, which are securely included in the given voucher, as HATEOAS propositions.

4 Validation and Illustration

As a validation of the principles and architecture of the voucher-based authentication and authorisation in a BPO context, we prototyped an accounting workflow with an outsourced billing workflow on top of the WF-Interop middleware [2]. The goal of this validation is to illustrate a decentralised, cross-organisational authorisation mechanism to support long-term remote interactions between heterogeneous workflow technologies.

To implement the accounting case study, we employed the jBPM workflow engine for the accounting process and the Ruote workflow engine for the billing process. They communicate via the WF-Interop middleware (see Fig. 6). Figure 5 illustrates a simplified accounting workflow responsible for starting a process at a

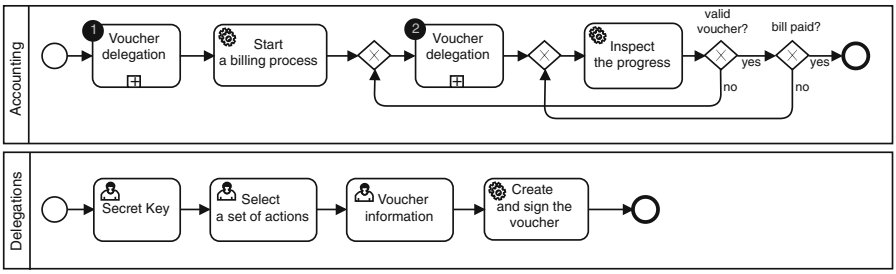

Fig. 5. Outsourcing a billing process in jBPM workflow engine.

billing provider (not shown in the figure) and inspecting the progress periodically, as well as a subprocess for creation or delegation of the vouchers.

The accountant is the business owner holding a voucher (V_a). (1) He delegates some privileges to the billing provider by creating a voucher (V_{ab}) (e.g. by adding a claim enabling the billing provider to start a process at his trusted package delivery service.). Then he starts a billing process at the billing provider by providing V_a and V_{ab}. Using V_a WF-Interop authorises the action and passes the V_{ab} to the billing provider for further BPO interactions in the future. (2) In the second stage, the client workflow engine is responsible to track the progress of the billing process which may take weeks. To start the periodic progress inspection, the workflow engine needs to have a voucher in order to be authorised against WF-Interop. Therefore, the accountant delegates the progress inspection privilege as a claim by creating a voucher V_{ae} for the responsible workflow engine. Then the workflow engine performs the inspection by passing V_{ae} to WF-Interop. As long as the voucher is still valid (not rejected by WF-Interop because of the validity period or voucher revocation), it inspects the progress. As soon as the voucher is expired, the voucher must be renewed by the accountant.

The main goal of this validation is to show (i) that the BPO client (accounting workflow) creates a voucher for delegation of a subset of their resources to the BPO provider for further usage; (ii) hierarchical, intra-organisational voucher delegations among entities (e.g. between the accountant and the workflow engine); (iii) secure asynchronous and impersonated operations when the business owner is offline (e.g. the periodic progress inspections by the workflow engine when the accountant is not logged in). (iv) The WF-Interop HATEOAS propositions within responses (not shown in the figure) are limited to the existing claims inside the vouchers (e.g. upon each invocation, the workflow engine is informed that it can only execute the progress inspection).

Furthermore, WF-Interop performs the verification procedure on the delegated actions based on the given vouchers. In other words, it stores no information about the access rights of the delegatees in order to be *stateless* and to respect organisational *confidentiality* according to the specific structure of the vouchers as described in Sect. 3.1.

5 Performance Evaluation

In this subsection, we evaluate the performance impact of vouchers with varying number of delegations. More precisely, we evaluate the progress inspection requests on an outsourced process.

Set-up. To evaluate the performance overhead of vouchers, we implemented a security interceptor on top of the REST interfaces of the WF-Interop middleware. Since the voucher verification is the most recurring routine compared to the other activities, the progress inspection on the Ruote workflow engine is the target of this evaluation. Both the WF-Interop middleware and the security interceptor are written in Java using Spring Boot. The cryptographic schemes employed in the set-up for the hybrid encryption of parent vouchers are based on both RSA or Elliptic Curve (EC) cryptography in separate experiments, integrated with AES symmetric encryption algorithm. Voucher signatures are also implemented using RSA or EC[6].

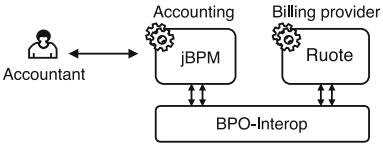

Fig. 6. Outsourcing a billing process.

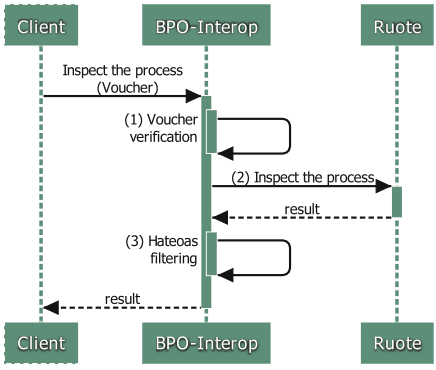

Fig. 7. Performance evaluation set-up.

As illustrated in Fig. 7, a client sends a request to a Ruote workflow engine via the WF-Interop middleware to inspect the progress of a process. The procedure contains three sub-routines: (1) voucher verification, (2) process inspection and (3) HATEOAS filtering. We executed the procedure 500 times for different cases: using no voucher, as well as using vouchers with 1 to 6 delegations.

As illustrated in Fig. 8, the process inspection (baseline) has constant execution time of 5.27 ms. The HATEOAS filtering has low impact of maximum 2.99 ms with six delegations. Therefore, the main performance overhead is on

[6] The RSA-based hybrid encryption is based on RSA/OAEP with AES/GCM, SHA-256 and MGF1 padding. The EC-based hybrid encryption is based on ECIES with AES/CBC, HMAC-SHA256, KDF2 and ECSVDP-DH (Elliptic Curve Secret Value Derivation Primitive [7,16]). The RSA-based signature algorithms are based on RS256 and RS512 and the EC-based ones are ES256, ES384 and ES512 [15].

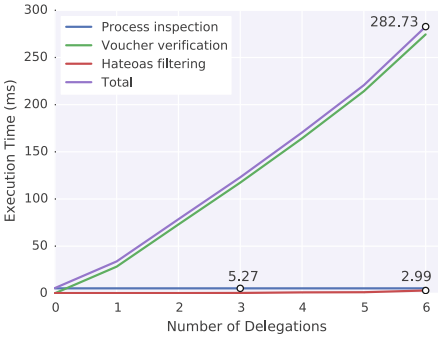

Fig. 8. The setup employs 4096 bits key size for RSA-based hybrid encryption and voucher signatures.

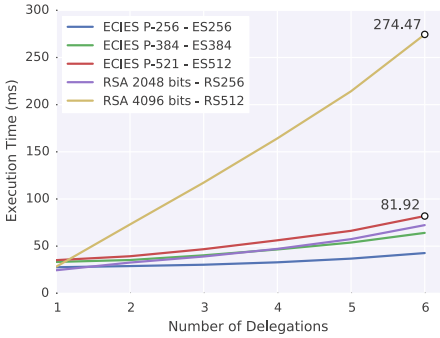

Fig. 9. Different setups employ various hybrid encryption and signature algorithms for voucher verification.

voucher verification. Considering a growing number of delegations, the signature verification of vouchers, as well as decryption and verification of parent vouchers are the main reasons for the performance overhead.

Results. The experiment results which are presented in Fig. 8 shows the worst-case scenario because we employed a set of strong encryption schemes with large key lengths. We also evaluated the overhead against the fastest WF-Interop function (namely process inspection). In a BPO context, these performance results can be considered low as the number of delegations rarely becomes six. The total response time of 282.73 ms is still acceptable in a BPO context.

We further evaluated the same experiments with different key lengths using RSA-based hybrid encryption, or Elliptic Curve integrated encryption scheme (ECIES) with various signature algorithms (see Fig. 9). The results show that the execution time of voucher verification is considerably improved from 274.47 ms to 81.92 ms in the case of 6 delegations. The detailed experiment results are provided in the appendix.

6 Conclusion and Future Work

In this paper, we presented a voucher-based authorisation architecture and middleware for BPO. Vouchers enable federated workflow systems to iteratively delegate all or subsets of claims to different entities. Nested vouchers protect the confidentiality of the parent issuers by using hybrid encryption, which is important in a cross-organisational context. Moreover, vouchers facilitate the asynchronous BPO operations in long-running workflows when the authorised subject has no active session in the workflow engine. We employed vouchers to filter the *Hypermedia as the Engine of Application State (HATEOAS)* propositions using an interceptor pattern in the WF-Interop middleware. We implemented a security module for the middleware and, on top of it, validated the architecture

by prototyping a billing workflow case study. Furthermore, the evaluation showed that the measured performance overhead is acceptable in a BPO context.

The current centralised approach suffers from a single point of failure issue, but on the other hand, some of the important security controls such as voucher revocation and context-sensitive delegations can practically be employed in the current design. Our future work includes moving towards a decentralised architecture, and addressing potential limitations such as key management issues, context-sensitive and fine-grained controls on delegations by resource owners in that setting.

A Appendix

See Tables 1 and 2.

Table 1. The setup employs 4096 bits key size for RSA-based hybrid encryption and voucher signatures. Execution times (ms) are as follows.

Number of delegations	0	1	2	3	4	5	6
Process inspection	5.27	5.27	5.27	5.27	5.27	5.27	5.27
Voucher verification	0	28.28	73.02	117.36	164.26	214.22	274.47
Hateoas filtering	0.27	0.27	0.27	0.39	0.99	1.16	2.99
Total	5.54	33.82	78.56	123.02	170.52	220.65	282.73

Table 2. Different setups employ various hybrid encryption and signature algorithms for voucher verification. Execution times (ms) are as follows.

Number of delegations	1	2	3	4	5	6
ECIES P-256 - ES256	27.63	28.89	30.31	32.87	36.79	42.68
ECIES P-384 - ES384	33.28	35.30	40.29	46.50	53.82	64.07
ECIES P-521 - ES512	35.19	39.30	46.77	56.21	66.26	81.92
RSA 2048 bits - RS256	24.48	32.61	39.02	47.13	57.45	72.30
RSA 4096 bits - RS512	28.28	73.02	117.36	164.26	214.22	274.47

References

1. ITU-T: Baseline identity management terms and definitions, X.1252 (2010)
2. Beni, E.H., Lagaisse, B., Joosen, W.: WF-Interop: adaptive and reflective rest interfaces for interoperability between workflow engines. In: Proceedings of the 14th International Workshop on Adaptive and Reflective Middleware, p. 1. ACM (2015)

3. Bertino, E., Ferrari, E., Atluri, V.: The specification and enforcement of authorization constraints in workflow management systems. ACM Trans. Inf. Syst. Secur. (TISSEC) **2**, 65–104 (1999)
4. Birgisson, A., Politz, J.G., Erlingsson, U., Taly, A., Vrable, M., Lentczner, M.: Macaroons: cookies with contextual caveats for decentralized authorization in the cloud (2014)
5. Coalition, W.M.: Terminology and glossary. WFMC Document WFMCTC-1011, Workflow Management Coalition, Avenue Marcel Thiry 204, 1200 (1996)
6. Dayasindhu, N.: Information technology enabled process outsourcing and reengineering: case study of a mortgage bank. In: AMCIS 2004 Proceedings, p. 437 (2004)
7. Diffie, W., Hellman, M.: New directions in cryptography. IEEE Trans. Inf. Theor. **22**(6), 644–654 (1976)
8. Ellison, C., Frantz, B., Lampson, B., Rivest, R., Thomas, B., Ylonen, T.: SPKI certificate theory (IETF RFC 2693) (1999)
9. Ferraiolo, D., Cugini, J., Kuhn, D.R.: Role-based access control (RBAC): Features and motivations. In: Proceedings of 11th Annual Computer Security Application Conference, pp. 241–48 (1995)
10. Ford, M., Endpoints, A., Keller, C., Kloppmann, M., König, D., Leymann, F., Müller, R., Pfau, O.G.: Web services human task (WS-HumanTask), v1.0 (2007)
11. Halvey, J.K., Melby, B.M.: Business Process Outsourcing: Process, Strategies, and Contracts. Wiley, New York (2007)
12. Hardt, D.: The OAuth2 authorization framework (2012)
13. Hu, V.C., Ferraiolo, D., et al.: Guide to attribute based access control (ABAC) definition and considerations (draft). NIST Special Publication 800(162) (2013)
14. Jones, M., Bradley, J., Sakimura, N.: JSON web signature (JWS). Technical report (2015)
15. Jones, M., Bradley, J., Sakimura, N.: JSON web token (JWT). Technical report (2015)
16. Koblitz, N.: Elliptic curve cryptosystems. Math. Comput. **48**(177), 203–209 (1987)
17. Mettraux, J., Kalmer, K., Meyers, R., de Mik, H., Kohlbecker, A., et al.: Ruote-a ruby workflow engine
18. Muller, J., Mulle, J., von Stackelberg, S., Bohm, K.: Secure business processes in service-oriented architectures-a requirements analysis. In: 2010 IEEE 8th European Conference on Web Services (ECOWS), pp. 35–42. IEEE (2010)
19. Nadalin, A., Goodner, M., Gudgin, M., Barbir, A., Granqvist, H.: Oasis WS-Trust 1.4. Specification Version 1, pp. 41–45 (2008)
20. Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., Mortimore, C.: Openid connect core 1.0. The OpenID Foundation, p. S3 (2014)
21. Schefer-Wenzl, S., Bukvova, H., Strembeck, M.: A review of delegation and break-glass models for flexible access control management. In: Abramowicz, W., Kokkinaki, A. (eds.) BIS 2014. LNBIP, vol. 183, pp. 93–104. Springer, Cham (2014). doi:10.1007/978-3-319-11460-6_9
22. Van Alsenoy, B., De Cock, D., Simoens, K., Dumortier, J., Preneel, B.: Delegation and digital mandates: legal requirements and security objectives. Comput. Law Secur. Rev. **25**(5), 415–431 (2009)
23. Wainer, J., Barthelmess, P., Kumar, A.: W-RBAC a workflow security model incorporating controlled overriding of constraints. Int. J. Coop. Inf. Syst. **12**(04), 455–485 (2003)
24. Wainer, J., Kumar, A.: A fine-grained, controllable, user-to-user delegation method in RBAC. In: Proceedings of the Tenth ACM Symposium on Access Control Models and Technologies, pp. 59–66. ACM (2005)

25. Wüllenweber, K., Beimborn, D., Weitzel, T., König, W.: The impact of process standardization on business process outsourcing success. Inf. Syst. Front. **10**(2), 211–224 (2008)
26. Zhang, L., Ahn, G.J., Chu, B.T.: A rule-based framework for role-based delegation and revocation. ACM Trans. Inf. Syst. Secur. (TISSEC) **6**(3), 404–441 (2003)